Remarks

In the present response, no claims are amended. Claims 1-8 and 10-23 are presented for examination.

Claim Rejections: 35 USC § 102 & § 103

Claims 1-7, 16, 17, and 21-23 are rejected under 35 USC § 102(e) as being anticipated by USPN 6,134,710 (hereinafter Levine). Claims 8-15 and 18 are rejected under 35 USC § 103(a) as being unpatentable over Levine in view of USPN 6,134,710 (Eickemeyer). Claims 19 and 20 are rejected under35 USC § 103 as being unpatentable over Levine in view of Grimsrud. These rejections are traversed.

The independent claims recite numerous recitations that are not taught or even suggested in the art of record. By way of example, claim 1 recites "replacing the selected instruction in the object code with a **break instruction**"(emphasis added). This recitation clearly distinguishes over Levine. Claim 1 further recites "resuming execution of the object code **without changing addresses of subsequent instructions in the object code**." This recitation also clearly distinguishes over Levine.

As explained by inventor Alan Karp in the interview of April 19, 2006, FIG. 12 of Levine teaches that branch instructions (not break instructions) are inserted into the object code. The branch instructions in Levine, however, alter the addresses of subsequent instructions in the object code when the object code resumes executing. Branch instructions function very differently than break instructions. Nowhere does Levine teach or even suggest inserting "break instruction" in the object code. Further, when Levine inserts branch instructions into the object code, these instructions change addresses of subsequent instructions in the object code. This teaching is contrary to the recitations of claim 1. Again, claim 1 recites "resuming execution of the object code without changing addresses of subsequent instructions in the object code."

For at least these reasons, independent claim 1 and its dependent claims are allowable over Levine and the art of record. Independent claims 8 and 16 recite similar recitations. Specifically, claim 8 recites an object code adapter that "inserts a break instruction into the object code, the break instruction replacing a selected instruction in the object code" and then resuming "execution of the object code without changing addresses of subsequent instructions in the object code." Claim 16 recites "replacing a selected instruction in the object code with a break instruction." Claim 16 then recites "resuming, upon execution of the instruction to resume, execution of the object code without changing addresses of subsequent instructions in the object code."

Rebuttal to Response to Arguments

On pages 12 and 13 of the Office Action, the Examiner provides a response to Applicants' claims. The Examiner argues that Levine's system of providing branch instructions "performs all the functions performed by applicant's break instruction" (see OA at p. 13). The Examiner also argues that the term "break instruction" in claims 1, 8, and 16 is indefinite because the accepted meaning of this term is "to interrupt execution at a given spot." Applicants traverse.

The term break instruction does mean "to interrupt execution at a given spot." A break instruction, however, does not "cause the processor to freeze or stall executions." That action would be nonsensical with current computer processors.[1] By contrast, a break instruction invokes a trap handler in the operating system (OS) kernel. This trap handler performs many functions. For example, trap handlers can branch to a debugger. As another example (as in exemplary embodiments of our invention), the trap handler branches to hint code.

One simply cannot use a conventional branch instruction of Levin as a substitution for a break instruction in our claims. A branch instruction needs an op code and a branch target. Some machines have instructions that consist of only an op code. Where could we put the branch target without changing the locations of subsequent instructions on such a machine? Further, every machine has a non-obtrusive break instruction that fits in the space of the smallest sized instruction the machine supports.

A branch instruction has a branch target as part of the instruction. That target may be an offset from the current location encoded as bits in the instruction itself or an

[1] One skilled in the art knows that on very old processors (such as the single program machines, like the IBM 7090), a break instruction would case the processor to freeze. On those machines the user could sit at the console and single step the program. The only way to recover would be to reboot the machine. This technology has been outdated and not used for many years.

address contained in a register specified in the branch instruction. A break instruction is a special kind of branch, one that invokes a trap handler in the OS kernel. By stark contrast, break instructions do not need a branch target because that is specified in the interrupt table in the OS that has an address to branch to for each kind of trap. One skilled in the art uses the term "break" rather than a "branch" because of this difference.

By way of example, claim 16 recites that the break instruction causes the processor to retrieve hint code. This step is true and accurate. As one skilled in the art knows, a break instruction invokes the trap handler in the OS kernel. A look-up table is consulted to determine where to branch. In claim 16, the processor branches to hint code. After executing the hint code, the processor resumes execution of the same instruction set "without changing addresses of subsequent instructions in the object code." Applicants reiterate this point: If claim 16 were amended to recite "executing a branch instruction" instead of its current recitation (i.e., "executing a break instruction"), then execution of the branch instruction would **<u>alter</u>** addresses of subsequent instructions in the object code.

Applicants respectfully submit that the claims and description in the specification are in accord with current computer science terminology and knowledge of one skilled in the art.

Applicants have conducted an interview with the Examiner and included the inventor (Alan Karp: a principal scientist at Hewlett-Packard and a recognized expert in this technology). Mr. Karp has explained to the Examiner the difference between branch and break instructions and other shortcomings of Levine. Applicants have also filed an RCE and amended the claims to clearly distinguish Levine. Applicants believe that no further amendments are necessary to distinguish the art of record. Applicants respectfully ask for a Notice of Allowance to avoid an appeal.

CONCLUSION

In view of the above, Applicants believe that all pending claims are in condition for allowance. Allowance of these claims is respectfully requested.

Any inquiry regarding this Amendment and Response should be directed to Philip S. Lyren at Telephone No. 832-236-5529. In addition, all correspondence should continue to be directed to the following address:

Hewlett-Packard Company
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

Respectfully submitted,

/Philip S. Lyren #40,709/

Philip S. Lyren
Reg. No. 40,709
Ph: 832-236-5529